# Software Engineering Concepts
# INTE 31253

*BSc .(Honours) in Management & Information Technology*

*Department of Industrial Management*

*University of Kelaniya*

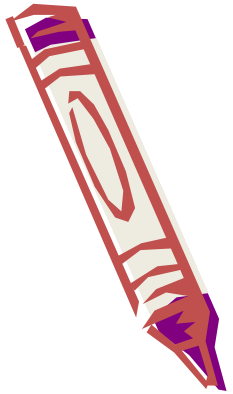
*Academic Year 2015/2016*

*Semester 1*

## Lecture 1: Introduction

Dilani Wickramaarachchi
dilani@kln.ac.lk

# Lectures



- **Look** - at what is being shown
- **Listen** - to what is being said
- **Think** - about the examples
- **Ask** - questions when you do not understand
- **Write** - when you think it is important
- ***DO NOT TALK AMONGST YOURSELVES***

# Lecture Overview

- Introduction to the Module
- Discuss evaluation criteria
- Software vs Hardware
- Software Development Life Cycle
- Software Engineering
- Software Applications and its Evolution
- Polya's four-step approach to problem solving
- Information flow in Software Projects
- Software Project Success- Standish Group 2015 Chaos Report

# Learning Outcomes of the module

- Define what software is
- List software development process models
- Recognize the need of formal approach for software development,
- Identify software requirements of an organization,
- Propose a suitable software solution for an organization,
- Select suitable software development process model, and tools for software development,
- Analyze and design software systems for organizations,
- Review software designs,
- Inspect software systems

# Course content

- Software and Software Engineering: Software process, project management, managing people, software cost estimation, quality management, configuration management, software requirements, system models, software prototyping, formal specification, architectural design, object-oriented design, user interface design, software testing,

# Evaluation Criteria

- Continuous assessment 40%
  - Case Discussion/Class presentation/Mid term exam 10
  - Group Assignment 30
- Final examination 60%
- At least 20% marks from both components are required to pass the module.

If you hand over a late assignment, the following criteria will be used to subtract points from your grade.

| | | **Total** |
|---|---|---|
| One day | - 10 points | 10 points |
| Two days | - 20 points | 30 points |
| Three days | - 30 points | 60 points |
| Four days | - 40 points | 100 points |

**After three days you will not get any points**

# Software Product

- Different from traditional types of products
  - intangible
    - difficult to describe and evaluate
  - malleable
  - human intensive
    - involves only trivial “manufacturing” process

# Software-Its Nature & Qualities

- Software engineering (SE) is an intellectual activity and thus human-intensive
- Software is built to meet a certain goal and satisfy certain qualities
- Software processes also must meet certain qualities
- Software qualities are sometimes referred to as “ilities”

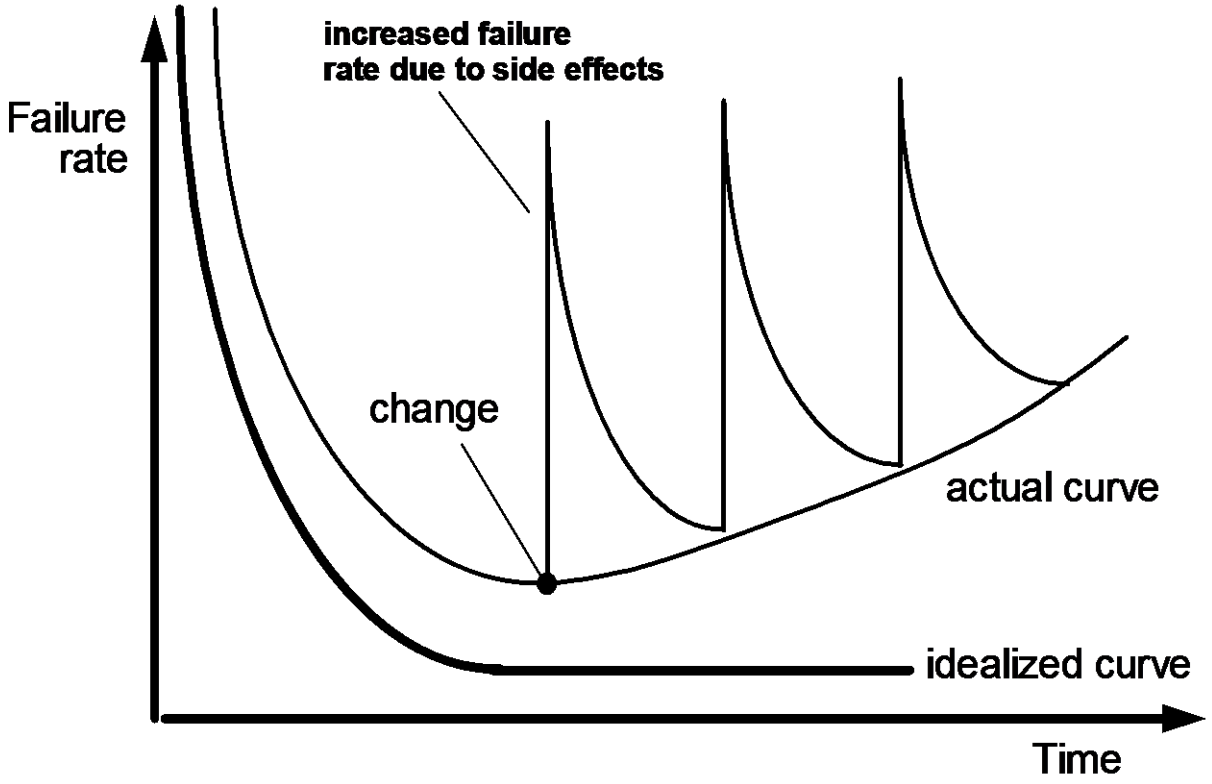
# Classification of sw qualities "ilities"

- Internal vs. external
  - External→ visible to users
  - Internal→ concern developers
- Product vs. process
  - Goal is to develop software products
  - The process is how we do it
- Internal qualities affect external qualities
- Process quality affects product quality

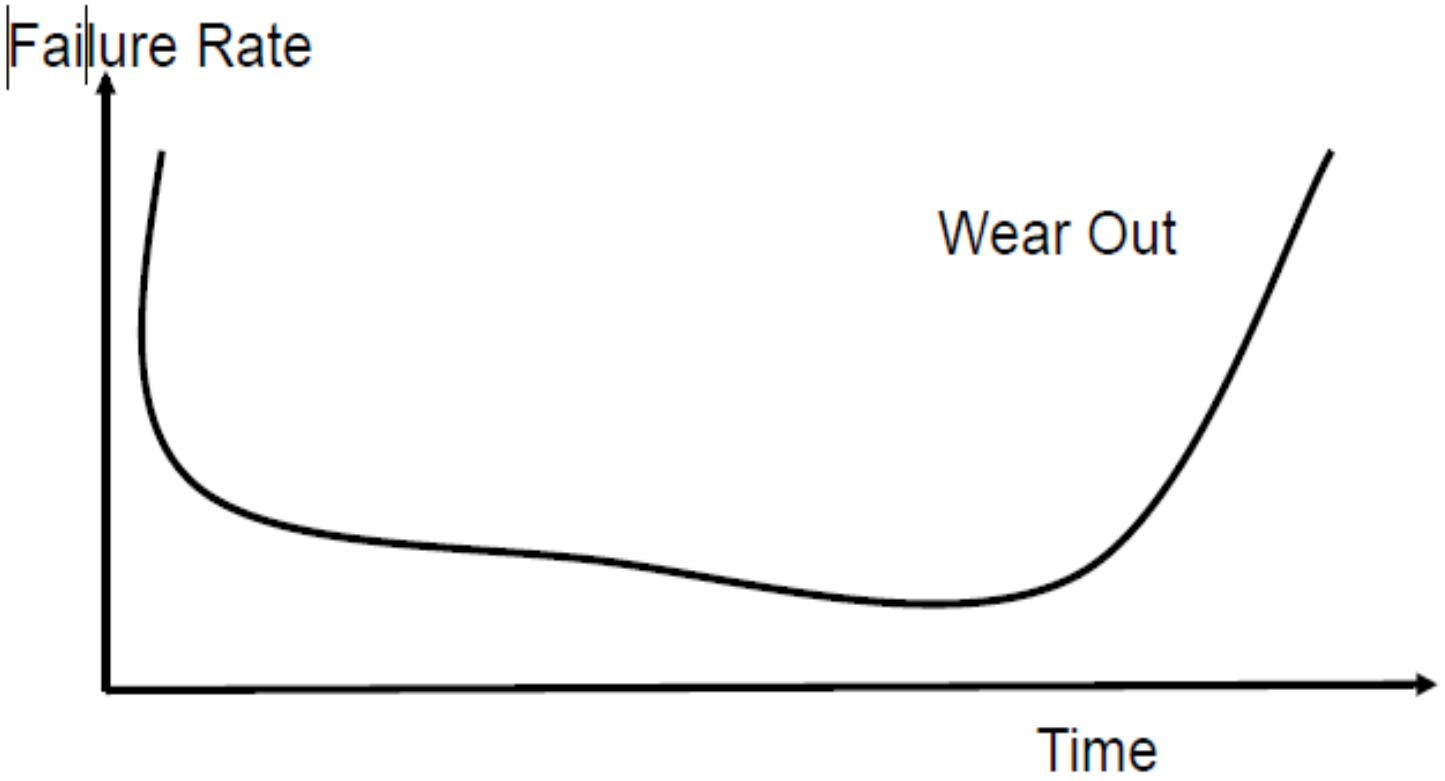
# Software vs Hardware


- Software – how it differs..
  - Software is developed or engineered, it is not manufactured in the classical sense.
  - Software doesn't "wear out."
  - Labour requirement

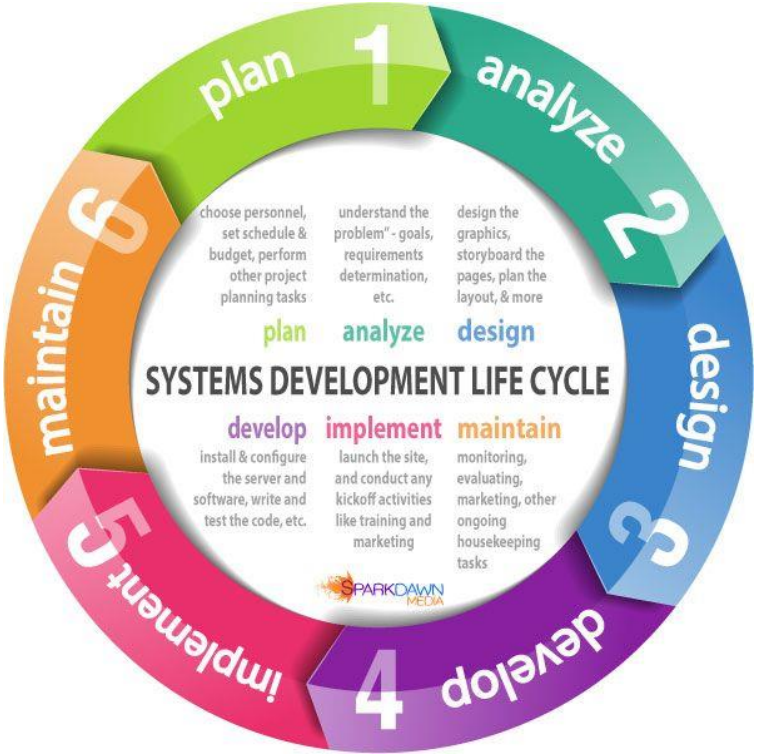
# Software-Failure Rate vs Time



# Hardware-Failure Rate vs Time



# Software Development life Cycle (SDLC)




# Why Software Engineering?

**A naive view:**

Problem Specification —*coding*→ Final Program

*But ...*

- Where did the *specification* come from?
- How do you know the specification corresponds to the *user's needs*?
- How did you decide how to *structure* your program?
- How do you know the program actually *meets the specification*?
- How do you know your program will always *work correctly*?
- What do you do if the users' *needs change*?
- How do you *divide tasks up* if you have more than a one-person team?

# What is Software Engineering? (I)

***Some Definitions and Issues***

*"state of the art of developing quality software on time and within budget"*

- Trade-off between perfection and physical constraints
  - SE has to deal with real-world issues
- State of the art!
  - Community decides on "best practice" + life-long education

# What is Software Engineering? (II)

*"multi-person construction of multi-version software"*

— Parnas

- Team-work
  - Scale issue ("program well" is not enough) + Communication Issue
- Successful software systems must evolve
  - Change is the norm, not the exception

# What is Software Engineering? (III)

*"software engineering is different from other engineering disciplines"*

— Sommerville

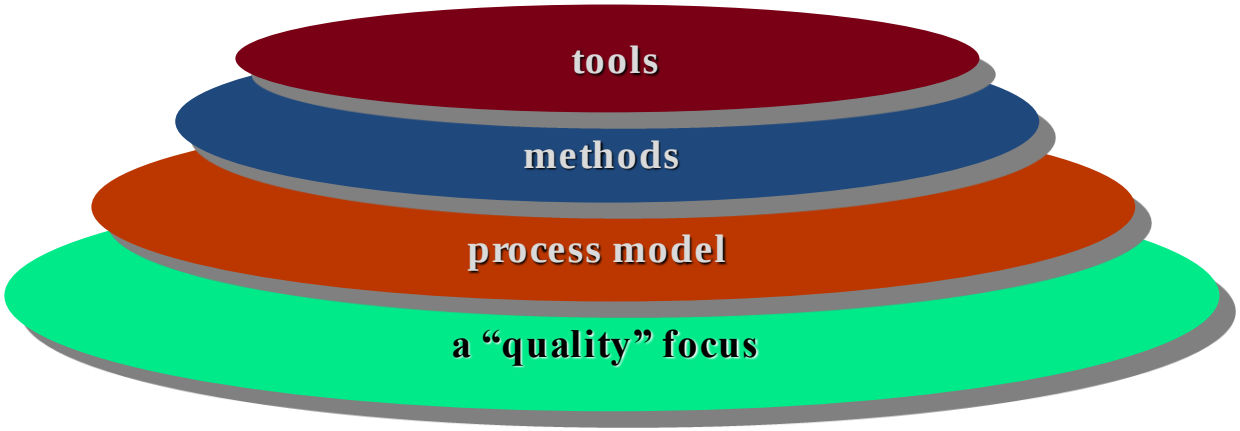
- Not constrained by physical laws
  - limit = human mind
- It is constrained by political forces
  - balancing stake-holders

# Software Engineering

- The seminal definition:
  - [Software engineering is] the establishment and use of sound engineering principles in order to obtain economical software that is reliable and works efficiently on real machines.
- The IEEE definition:

  *The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.*

# Software Engineering: A Layered Technology




*Software Engineering*

# Scientist vs Engineer

- Computer Scientist
  - Proves theorems about algorithms, designs languages, defines knowledge representation schemes
  - Has infinite time
- Engineer
  - Develops a solution for an application-specific problem for a client
  - Uses computers & languages, tools, techniques and methods
- Software Engineer
  - Works in multiple application domains
  - Has only 3 months….
  - …while changes occurs in requirements and available technology

# Software Applications

- System software
- Application software
- Engineering/scientific software
- Embedded software
- Product-line software
- Web applications
- BI software and Big Data
- IoT


# Software-should it change?.. Legacy to IoT

- software must be **adapted** to meet the needs of ***new computing environments or technology.***
- software must be **enhanced** to implement ***new business requirements.***
- software must be **extended to make it interoperable** with other more ***modern systems or databases***.
- software must be **re-architected** to make it viable within a network environment.

# Polya's four-step approach to problem solving

- Preparation: Understand the problem (communication and analysis).
- Thinking Time: Devise a plan (modeling and software design).
- Insight: Carry out the plan (code generation).
- Examine the result for accuracy (testing and quality assurance).

# Understand the Problem

- *Who has a stake in the solution to the problem?* That is, who are the stakeholders?
- *What are the unknowns?* What data, functions, and features are required to properly solve the problem?
- *Can the problem be compartmentalized?* Is it possible to represent smaller problems that may be easier to understand?
- *Can the problem be represented graphically?* Can an analysis model be created?

# Plan the Solution

- *Have you seen similar problems before?* Are there patterns that are recognizable in a potential solution? Is there existing software that implements the data, functions, and features that are required?
- *Has a similar problem been solved?* If so, are elements of the solution reusable?
- *Can subproblems be defined?* If so, are solutions readily apparent for the subproblems?
- *Can you represent a solution in a manner that leads to effective implementation?* Can a design model be created?

# Carry Out the Plan

- *Does the solution conform to the plan?* Is source code traceable to the design model?

- *Is each component part of the solution provably correct?* Has the design and code been reviewed, or better, have correctness proofs been applied to algorithm?

# Examine the Result

- *Is it possible to test each component part of the solution?* Has a reasonable testing strategy been implemented?

- *Does the solution produce results that conform to the data, functions, and features that are required?* Has the software been validated against all stakeholder requirements?

# Information flow in SW projects

How the customer explained it

How the Project Leader understood it

How the Software architect designed it

How the Programmer wrote it

How theTester left it

How the Business Consultant described it

How the project was documented

How the customer was billed

How it was supported

What the customer really needed

Source: unknown

DAAD project „Joint Course on Software Engineering" ©

17

# Standish Group 2015 Chaos Report

**MODERN RESOLUTION FOR ALL PROJECTS**

| | 2011 | 2012 | 2013 | 2014 | 2015 |
|---|---|---|---|---|---|
| SUCCESSFUL | 29% | 27% | 31% | 28% | 29% |
| CHALLENGED | 49% | 56% | 50% | 55% | 52% |
| FAILED | 22% | 17% | 19% | 17% | 19% |

*The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011–2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.*

# Success of a Software Project

- Project Management Institute (PMI) has defined success as onTime, onBudget, and onTarget also known as the *Triple Constraints*
- Six factors now being included in the overall measure of success
  - on Time, On Budget, on Target, on Goal, Value and Satisfaction

- project-based work vs product-based activities with stable teams

## CHAOS FACTORS OF SUCCESS

| FACTORS OF SUCCESS | POINTS | INVESTMENT |
|---|---|---|
| Executive Sponsorship | 15 | 15% |
| Emotional Maturity | 15 | 15% |
| User Involvement | 15 | 15% |
| Optimization | 15 | 15% |
| Skilled Resources | 10 | 10% |
| Standard Architecture | 8 | 8% |
| Agile Process | 7 | 7% |
| Modest Execution | 6 | 6% |
| Project Management Expertise | 5 | 5% |
| Clear Business Objectives | 4 | 4% |

# Bad SE practices create ..

- Failed projects
- Lost money
- Stressed employees
- Poor customer value

| YEAR | COMPANY | OUTCOME (COSTS IN US $) |
|---|---|---|
| 2005 | Hudson Bay Co. [Canada] | Problems with inventory system contribute to $33.3 million* loss. |
| 2004–05 | UK Inland Revenue | Software errors contribute to $3.45 billion* tax-credit overpayment. |
| 2004 | Avis Europe PLC [UK] | Enterprise resource planning (ERP) system canceled after $54.5 million† is spent. |
| 2004 | Ford Motor Co. | Purchasing system abandoned after deployment costing approximately $400 million. |
| 2004 | J Sainsbury PLC [UK] | Supply-chain management system abandoned after deployment costing $527 million.† |
| 2004 | Hewlett-Packard Co. | Problems with ERP system contribute to $160 million loss. |

# List of failed and overbudget custom software projects

https://en.wikipedia.org/wiki/List_of_failed_and_overbudget_custom_software_projects

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 2008 | 2013 | Digital Media Initiative | Digital production, media asset management | United Kingdom | State broadcaster | By 2013, the project was judged to be obsolete (as much cheaper commercial off the shelf alternatives by then existed) and was scrapped by BBC management. The BBC Director General said it had been a huge waste of money.[6] | more than £98m *(£81.7m)* | Outsourced, then insourced, then outsourced again | Cancelled |
| 2009 | 2013 | The Surrey Integrated Reporting Enterprise Network (SIREN) | Crime & criminal intelligence logging system | United Kingdom (Surrey) | Police Force | Not fit for purpose[7] | £14.8m | Outsourced | Scrapped |
| 2011 | 2014 | Pust Siebel | Police case management | Sweden | Police | Poor functioning, inefficient in work environments.[8] | SEK 300m ($35m)[9] | Outsourced | Scrapped |
| 2012 | 2014 | Cover Oregon | Healthcare exchange website | United States | State government | Site was never able to accept online enrollments, so users were instructed to mail in paper enrollments instead. | approx $200m | Outsourced | Cancelled, then client and supplier both sued each other |

# Some Frequently Cited Factors

- Underestimation of complexity
- Failure to establish appropriate control over requirements and/or scope
- Inadequate communication
- Failure to engage stakeholders
- Inadequate testing
- Lack of oversight or poor project management
- Poor quality implementations
- Lack of risk management
- Failure to specify/address performance requirements
- Poorly planned/managed transitions
- Excessive process to prevent previous problems

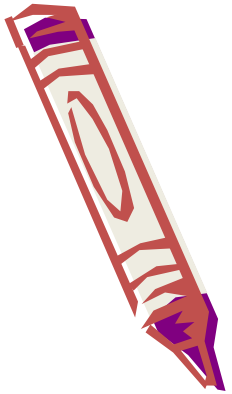
# Good SE practices create ..

- Successful projects
- Happy customers
- Business value
- Lower stress levels for developers


# Main objective of this Module

- Use good software engineering practices and learn to develop successful software

# End of the Lecture 1 -




*Go back to Lecture Overview*